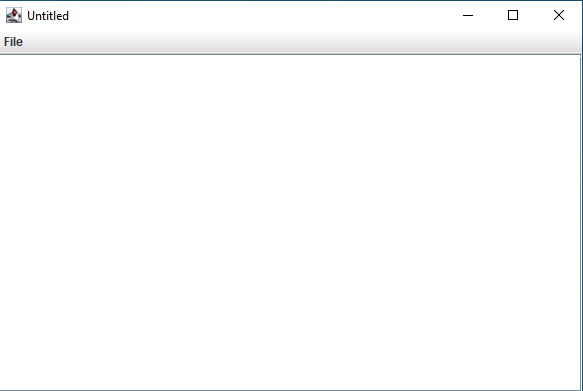
# Practical 5

The code compiling and running.



Using Notepad with exncryption

**package** notepad;

**import** java.security.InvalidAlgorithmParameterException;

**import** java.security.InvalidKeyException;

**import** java.security.Key;

**import** java.security.NoSuchAlgorithmException;

**import** java.security.NoSuchProviderException;

**import** java.security.Security;

**import** javax.crypto.BadPaddingException;

**import** javax.crypto.Cipher;

**import** javax.crypto.IllegalBlockSizeException;

**import** javax.crypto.NoSuchPaddingException;

**import** javax.crypto.spec.IvParameterSpec;

**import** javax.crypto.spec.SecretKeySpec;

**import** org.bouncycastle.jce.provider.BouncyCastleProvider;

**import** org.bouncycastle.util.encoders.Hex;

**import** java.awt.Dimension;

**import** java.awt.event.ActionEvent;

**import** java.awt.event.KeyAdapter;

**import** java.awt.event.KeyEvent;

**import** java.io.IOException;

**import** java.nio.charset.StandardCharsets;

**import** java.nio.file.Files;

**import** java.nio.file.Path;

**import** java.nio.file.Paths;

**import** java.util.concurrent.ExecutionException;

**import** javax.swing.AbstractAction;

**import** javax.swing.JFileChooser;

**import** javax.swing.JFrame;

**import** javax.swing.JMenu;

**import** javax.swing.JMenuBar;

**import** javax.swing.JOptionPane;

**import** javax.swing.JScrollPane;

**import** javax.swing.JTextArea;

**import** javax.swing.SwingUtilities;

**import** javax.swing.SwingWorker;

**import** javax.swing.WindowConstants;

@SuppressWarnings("serial")

**public** **class** Notepad **extends** JFrame {

**private** **static** **final** String ***UNTITLED*** = "Untitled";

**private** **final** JFileChooser fileChooser = **new** JFileChooser();

**private** **final** JTextArea textArea = **new** JTextArea();

**private** **boolean** modified = **false**;

**final** Cipher cipher = Cipher.*getInstance*("AES/CBC/PKCS7Padding", "BC");

**final** **byte**[] iv = Hex.*decode*("9f741fdb5d8845bdb48a94394e84f8a3");

**private** Key key = "";

**public** Notepad() {

**final** JMenu fileMenu = **new** JMenu("File");

fileMenu.add(**new** FileAction("New", **this**::newFile));

fileMenu.add(**new** FileAction("Open", **this**::openFile));

fileMenu.add("Save").addActionListener(event -> saveFile());

fileMenu.addSeparator();

fileMenu.add(**new** FileAction("Exit", () -> System.*exit*(0)));

**final** JMenuBar menuBar = **new** JMenuBar();

menuBar.add(fileMenu);

setJMenuBar(menuBar);

textArea.addKeyListener(**new** KeyAdapter() {

@Override

**public** **void** keyTyped(KeyEvent e) {

modified = **true**;

}

});

add(**new** JScrollPane(textArea));

setTitle(***UNTITLED***);

setPreferredSize(**new** Dimension(600, 400));

pack();

}

**private** **void** newFile() {

setTitle(***UNTITLED***);

textArea.setText(**null**);

modified = **false**;

}

// Opening a file can be a time-consuming task. Therefore, we open it in a

// worker thread.

**private** **class** FileOpener **extends** SwingWorker<**byte**[], Void> {

**private** **final** Path path;

**public** FileOpener(**final** Path path) {

**this**.path = path;

}

@Override

**protected** **byte**[] doInBackground() **throws** IOException {

**return** Files.*readAllBytes*(path);

}

@Override

**protected** **void** done() {

**try** {

**byte**[] data = get();

*decrypt*(cipher,key,iv,data);

textArea.setText(**new** String(data, StandardCharsets.***UTF\_8***));

setTitle(path.toString());

modified = **false**;

} **catch** (**final** InterruptedException | ExecutionException e) {

e.printStackTrace();

}

}

}

**private** **void** openFile() {

String alert = JOptionPane.*showInputDialog*("Input Key: ", key);

**final** **int** choice = fileChooser.showOpenDialog(**this**);

**if** (choice == JFileChooser.***APPROVE\_OPTION***) {

(**new** FileOpener(fileChooser.getSelectedFile().toPath())).execute();

}

}

// Saving a file can be a time-consuming task. Therefore, we save it in a

// worker thread.

**private** **class** FileSaver **extends** SwingWorker<Void, Void> {

**private** **final** Path path;

**private** **final** String text;

**public** FileSaver(**final** Path path, **final** String text) {

**this**.path = path;

**this**.text = text;

}

@Override

**protected** Void doInBackground() **throws** IOException {

**byte**[] data = text.getBytes(StandardCharsets.***UTF\_8***);

Files.*write*(path, data);

*encrypt*(cipher, key, iv, data);

**return** data;

}

@Override

**protected** **void** done() {

setTitle(path.toAbsolutePath().toString());

modified = **false**;

}

}

**public** **void** saveFile() {

Path path = **null**;

String key = "";

String alert = JOptionPane.*showInputDialog*("Insert key: ", key);

**if** (getTitle().equals(***UNTITLED***)) {

**int** choice = fileChooser.showSaveDialog(**this**);

**if** (choice == JFileChooser.***APPROVE\_OPTION***) {

path = fileChooser.getSelectedFile().toPath();

} **else** {

**return**;

}

} **else** {

path = Paths.*get*(getTitle());

}

(**new** FileSaver(path, textArea.getText())).execute();

}

**private** **class** FileAction **extends** AbstractAction {

**private** Runnable action;

**public** FileAction(String name, Runnable action) {

**super**(name);

**this**.action = action;

}

@Override

**public** **void** actionPerformed(**final** ActionEvent e) {

**if** (modified) {

**int** choice = JOptionPane.*showConfirmDialog*(Notepad.**this**,

"The text in " + getTitle()

+ " has changed\nDo you want to save it?",

"Notepad", JOptionPane.***YES\_NO\_CANCEL\_OPTION***,

JOptionPane.***WARNING\_MESSAGE***);

**switch** (choice) {

**case** JOptionPane.***YES\_OPTION***:

saveFile();

action.run();

**break**;

**case** JOptionPane.***NO\_OPTION***:

action.run();

**default**:

// cancel

}

} **else** {

action.run();

}

}

}

**public** **static** **void** main(**final** String[] args) {

SwingUtilities.*invokeLater*(() -> {

**final** Notepad notepad = **new** Notepad();

notepad.setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);

notepad.setVisible(**true**);

});

}

**private** **static** **byte**[] encrypt(**final** Cipher cipher, **final** Key key,

**final** **byte**[] initialisationVector, **final** **byte**[] data)

**throws** InvalidKeyException, InvalidAlgorithmParameterException,

IllegalBlockSizeException, BadPaddingException {

cipher.init(Cipher.***ENCRYPT\_MODE***, key,

**new** IvParameterSpec(initialisationVector));

**return** cipher.doFinal(data);

}

**private** **static** **byte**[] decrypt(**final** Cipher cipher, **final** Key key,

**final** **byte**[] initialisationVector, **final** **byte**[] data)

**throws** InvalidKeyException, InvalidAlgorithmParameterException,

IllegalBlockSizeException, BadPaddingException {

cipher.init(Cipher.***DECRYPT\_MODE***, key,

**new** IvParameterSpec(initialisationVector));

**return** cipher.doFinal(data);

}

}